Guideline: Building and Managing Custom Data Connectors (ETL Pipeline) in Python

# 1. Setting Up the Connector Environment

* a. Choose Your API Provider: Identify a data provider and understand its Base URL, Endpoints, and Authentication.
* b. Understand the API Documentation: Focus on headers, query params, pagination, rate limits, and response structure.

# 2. Secure API Authentication Using Environment Variables

* a. Create a `**.env`** File Locally: Store API keys and secrets as KEY=VALUE pairs.
* b. Load Environment Variables in Code: Use libraries like `dotenv` to securely load environment variables.

# 3. Design the ETL Pipeline

* Extract: Connect to the API, pass tokens/headers, and collect JSON data.
* Transform: Clean or reformat the data for MongoDB compatibility.
* Load: Store the transformed data into a MongoDB collection.

# 4. MongoDB Collection Strategy

* Use one collection per connector, e.g., `connector\_name\_raw`.
* Store ingestion timestamps to support audits or updates.

# 5. Iterative Testing & Validation

* Test for invalid responses, empty payloads, rate limits, and connectivity errors.
* Ensure consistent insertion into MongoDB.

# 6. Git and Project Structure Guidelines

* a. Use a Central Git Repository: Clone the shared repo and create a new branch for your connector.
* b. Ignore Secrets: Add `.env` to `.gitignore` before the first commit.
* c. Push and Document: Write README.md with endpoint details, API usage, and example output.

# Final Checklist for Students

* Understand API documentation
* Secure credentials in `.env`
* Build complete ETL script
* Validate MongoDB inserts
* Push code to your branch
* Include descriptive README
* Submit Pull Request